

## CONTROL SYSTEM USING PLURAL OBJECTS, CONSTRUCTING METHOD THEREFOR AND PERIPHERAL EQUIPMENT CONTROL SYSTEM

Patent Number: JP9106355  
Publication date: 1997-04-22  
Inventor(s): KIMURA YOSHIHIRO; HISAMATSU YUTAKA  
Applicant(s):: SEIKO EPSON CORP  
Requested Patent: ☐ JP9106355  
Application Number: JP19950264193 19951012  
Priority Number(s):  
IPC Classification: G06F9/46 ; G06F9/06 ; G06F9/44 ; G06F13/10  
EC Classification:  
Equivalents:

### Abstract

**PROBLEM TO BE SOLVED:** To more flexibly and easily provide a customizable control system and constructing method therefor, in the control system composed of plural common objects.

**SOLUTION:** A first OCX (object liking and embedding custom control) 10 to be the instance of a common object is provided with an interface 12 transmitting an event and an interface 13 receiving the event, in addition to an interface providing property and methods. The address of the interface 13 of the first OCX 10 is passed to the interface 12 of a second OCX 11 by an interface object 15 and the connection on the event is formed. Thus, the bi-directional communication between the OCX 10 and 11 becomes possible, the function of the second OCX is made the full use from the first OCX 10 and a system which is capable of obtaining a quick response can be constructed.

Data supplied from the esp@cenet database - I2

**THIS PAGE BLANK (USPTO)**

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-106355

(43) 公開日 平成9年(1997)4月22日

| (51)IntCl. <sup>9</sup> | 識別記号  | 庁内整理番号 | F I     | 技術表示箇所 |         |
|-------------------------|-------|--------|---------|--------|---------|
| G 0 6 F                 | 9/46  | 3 4 0  | G 0 6 F | 9/46   | 3 4 0 A |
|                         | 9/06  | 5 3 0  |         | 9/06   | 5 3 0 W |
|                         | 9/44  | 5 3 0  |         | 9/44   | 5 3 0 M |
|                         | 13/10 | 3 3 0  |         | 13/10  | 3 3 0 A |

審査請求 未請求 請求項の数 9 O L (全 16 頁)

(21) 出願番号 特願平7-264193

(22) 出願日 平成7年(1995)10月12日

(71) 出願人 000002369

セイコーエプソン株式会社

東京都新宿区西新宿2丁目4番1号

(72) 発明者 木村 芳弘

長野県諏訪市大和3丁目3番5号 セイコ

ーエプソン株式会社内

(72) 発明者 久松 豊

長野県諏訪市大和3丁目3番5号 セイコ

ーエプソン株式会社内

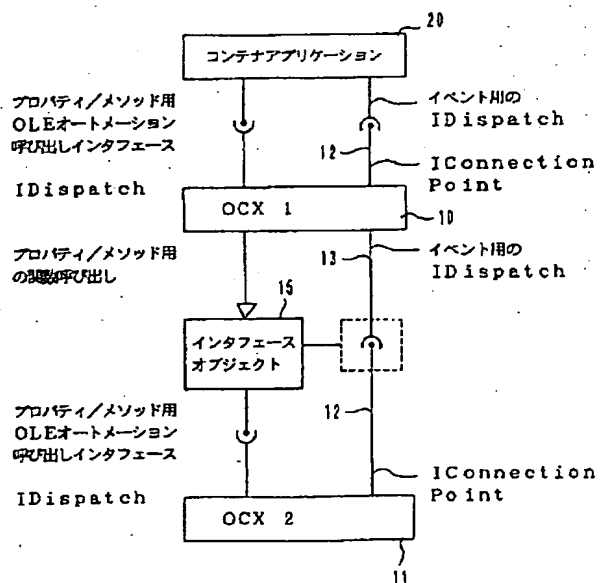
(74) 代理人 弁理士 鈴木 喜三郎 (外1名)

(54) 【発明の名称】 複数のオブジェクトを用いた制御システム、その構築方法および周辺装置制御システム

(57) 【要約】

【課題】 複数のコンモンオブジェクトから構成される制御システムにおいて、よりフレキシブルに、また、容易にカスタマイズ可能な制御システムおよびその構築方法を提供する。

【解決手段】 コンモンオブジェクトのインスタンスである第1のOCX 10にプロパティおよびメソッドを提供するインタフェースに加え、イベントを発信するインタフェース12とイベントを受信するインタフェース13とを設ける。そして、インタフェースオブジェクト15によって第1のOCX 10のインタフェース13のアドレスを第2のOCX 11のインタフェース12に渡し、イベントに関するコネクションを成立させる。これによって、OCX 10および11の間で双方向の通信が可能となり、第1のOCX 10から第2のOCXの機能をフルに活用し、迅速な応答の得られるシステムを構築できる。



## 【特許請求の範囲】

【請求項1】 属性値を含むプロパティおよびインプリメントされたファンクションを呼び出すメソッドの少なくともいずれかを提供する機能と、非同期に発生したアクションを含むイベントを発信する第1のインタフェースとを備えた複数の制御オブジェクトを有し、第1の前記制御オブジェクトまたはそのインスタンスによって第2の前記制御オブジェクトのインスタンスが作成され、前記第1の制御オブジェクトまたはそのインスタンスから前記第2の制御オブジェクトのインスタンスの制御が行われるコンピュータ内のシステムであって、前記第1の制御オブジェクトは、前記イベントを受信する第2のインタフェースを備えており、さらに、前記プロパティおよびメソッドの少なくともいずれかを前記第1の制御オブジェクトまたはそのインスタンスと前記第2の制御オブジェクトのインスタンスの間で伝達する機能と、前記第1の制御オブジェクトまたはそのインスタンスの前記第2のインタフェースおよび前記第2の制御オブジェクトのインスタンスの前記第1のインタフェースの少なくともいずれか一方の識別子を他方に渡す機能とを備えたインタフェースオブジェクトまたはそのインスタンスを有することを特徴とする複数のオブジェクトを用いた制御システム。

【請求項2】 請求項1において、前記第2の制御オブジェクトは複数の前記イベントを第2の配列に従って発信し、前記第1の制御オブジェクトは受信した前記複数のイベントの配列を第1の配列に変換する手段を備えていることを特徴とする複数のオブジェクトを用いた制御システム。

【請求項3】 属性値を含むプロパティおよびインプリメントされたファンクションを呼び出すメソッドの少なくともいずれかを提供する機能と、非同期に発生するアクションを含むイベントを発信する第1のインタフェースとを備えた複数の制御オブジェクトを有し、第1の前記制御オブジェクトまたはそのインスタンスによって第2の前記制御オブジェクトのインスタンスが作成され、前記第1の制御オブジェクトは前記イベントを受信する第2のインタフェースを備えており、さらに、前記プロパティおよびメソッドの少なくともいずれかを前記第1の制御オブジェクトまたはそのインスタンスと前記第2の制御オブジェクトのインスタンスとの間で伝達する機能と、前記第1の制御オブジェクトまたはそのインスタンスの前記第2のインタフェースおよび前記第2の制御オブジェクトのインスタンスの前記第1のインタフェースの少なくともいずれか一方の識別子を他方に渡す機能とを備えたインタフェースオブジェクトまたはそのインスタンスを有する制御システムの構築方法であって、

前記第1の制御オブジェクトまたはそのインスタンスが前記インタフェースオブジェクトのインスタンスを作成

するステップと、

前記インタフェースオブジェクトのインスタンスが前記第2の制御オブジェクトのインスタンスを作成するステップと、

前記第1の制御オブジェクトまたはそのインスタンスの前記第2のインタフェースと、前記第2の制御オブジェクトのインスタンスの前記第1のインタフェースとのコネクションを張るステップとを有することを特徴とする制御システムの構築方法。

10 【請求項4】 請求項3において、前記第1の制御オブジェクトは複数の前記イベントを受信する第1の配列を備えており、前記第2の制御オブジェクトは前記複数のイベントを発信する第2の配列を備えており、前記コネクションを張るステップにおいて、前記第2の配列を前記第1の配列に変換する第3の配列を作成することを特徴とする制御システムの構築方法。

【請求項5】 アプリケーションシステムに対し周辺装置の管理を行う制御システムであって、

属性値を含むプロパティおよびインプリメントされたファンクションを呼び出すメソッドの少なくともいずれかを提供する機能と、非同期に発生するアクションを含むイベントを発信する第1のインタフェースとを備えた複数の制御オブジェクトまたはそのインスタンスを有し、これら複数の制御オブジェクトまたはそのインスタンスからなる制御オブジェクト群は、前記イベントを受信する第2のインタフェースをさらに備えた第1の前記制御オブジェクトまたはそのインスタンスと、これら第1の制御オブジェクトまたはそのインスタンスによって作成された第2の前記制御オブジェクトのインスタンスと、

30 前記プロパティおよびメソッドの少なくともいずれかを前記第1の制御オブジェクトまたはそのインスタンスと前記第2の制御オブジェクトのインスタンスの間で伝達する機能と、前記第1の制御オブジェクトまたはそのインスタンスの前記第2のインタフェースおよび前記第2の制御オブジェクトのインスタンスの前記第1のインタフェースの少なくともいずれか一方の識別子を他方に渡す機能とを備えたインタフェースオブジェクトまたはそのインスタンスとを有していることを特徴とする周辺装置の制御システム。

【請求項6】 請求項5において、前記第2の制御オブジェクトは、前記周辺装置の第1の周辺装置に対応していることを特徴とする周辺装置の制御システム。

【請求項7】 請求項6において、前記第2の制御オブジェクトのプロパティは前記第1の周辺装置に固有な仕様に対応した属性値を含み、前記メソッドは前記第1の周辺装置に固有な命令を含み、さらに、前記イベントは前記第1の周辺装置において非同期に発生するアクションを反映可能であることを特徴とする周辺装置の制御システム。

【請求項8】 請求項6において、前記第1の制御オブジェクトは、前記インタフェースオブジェクトを介して伝達された前記第2の制御オブジェクトのプロパティ、メソッドおよびイベントの少なくともいずれかを普遍的な仕様に変換することを特徴とする周辺装置の制御システム。

【請求項9】 請求項6において、前記アプリケーションシステムはPOS制御システムであり、前記第2の制御オブジェクトはPOSを構成可能な各種の外部周辺装置毎に用意されており、前記POS制御システムは前記第1の制御オブジェクトまたはそのインスタンスを介して前記外部周辺装置を制御することを特徴とする周辺装置の制御システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、共通に使用できる複数のオブジェクトを備え、これらのオブジェクトを用いてカスタマイズ可能な制御システム、その構築方法および周辺装置制御システムに関するものである。

【0002】

【発明が解決しようとする課題】近年、コンピュータを用いて多種多様の処理が行われ、コンピュータに接続可能な外部入出力機器も多種多様なものが用意されている。従って、コンピュータを中心として、ユーザーの環境や目的などに合わせてフレキシブルに構築でき、カスタマイズ可能なシステムを提供することが重要となっている。例えば、共通な規格のバスを用いて1つのコンピュータに様々なタイプのプリンター、ディスプレイ、バーコードリーダーを始めその他の入出力機器が接続可能であり、コンピュータにはこれらの入出力機器を制御可能なソフトウェアがインストールされる場合には、これらの機器を的確に制御するように制御システムが構築されていなければならない。また、複数のコンピュータをネットワークで結んだ環境が簡単に構築できるので、個々のコンピュータによって処理すべき対象およびその手順を様々に変更できることが望ましい。

【0003】このようなフレキシブルなシステムをコンピュータ内に構築する1つの方法として、共通に用いることのできる複数のオブジェクト（コモンオブジェクト）を作成し、これらを活用してオペレーティングシステムやアプリケーションプログラムを構築することが考えられる。マイクロソフト社のOLE（Object Linking and Embedding）およびOLEプログラミング用に用意されたインタフェースを用いたOLEオートメーションあるいはOLEカスタムコントロール（OCX）といったシステムはその1つである。

【0004】本発明においては、コモンオブジェクトを活用して多種多様なオペレーティングシステムあるいはアプリケーションプログラムを簡単に構築できるシステ

ムおよびその方法を提供することを目的としている。特に、1つのコモンオブジェクトが他のコモンオブジェクトの機能を十分に引き出し得る制御システムおよびその構築方法を提供することを目的としている。さらに、多種多様な周辺装置の制御などに適したカスタマイズの容易なオープンなシステムでありながら処理速度の早い制御システムを提供することも本発明の目的の1つである。

【0005】

10 【課題を解決するための手段】本発明は、コモンオブジェクトとしての機能を備えた制御オブジェクト、すなわち、属性値を含むプロパティおよびインプリメントされたファンクションを呼び出すメソッドの少なくともいずれかをアプリケーションプログラムや他の制御オブジェクトあるいはそのインスタンスに提供する機能と、非同期に発生するアクションを含むイベントを発信する第1のインタフェースとを備えた制御オブジェクトに、さらに、イベントを受信する第2のインタフェースを設け、複数のこれら制御オブジェクトを用いて制御システムを構成するようにしている。

20 【0006】従って、本発明の制御システムでは、第1の制御オブジェクトまたはそのインスタンスはイベントを受信する第2のインタフェースを備えており、第2の制御オブジェクトのインスタンスにおいて非同期に発生するイベントを第1のインタフェースおよび第2のインタフェースを介して第1の制御オブジェクトまたはそのインスタンス、あるいは上流のアプリケーションプログラムに遅滞なく伝達できる。このため、第1の制御オブジェクトまたはそのインスタンスによって第2の制御オブジェクトのインスタンスを作成し、それらの制御を行うことによって多種多様な制御システムを簡単に構築でき、処理速度の早い多機能の制御システムとすることが

30 【0007】第1の制御オブジェクトまたはそのインスタンスの第2のインタフェースと、第2の制御オブジェクトのインスタンスの第1のインタフェースとの間にコネクションを張るためには、少なくともいずれか一方の識別子を他方に渡すことが必要である。本発明においては、プロパティおよびメソッドの少なくともいずれかを第1の制御オブジェクトまたはそのインスタンスと第2の制御オブジェクトのインスタンスの間で伝える機能と、上記の識別子を渡す機能とを備えたインタフェースオブジェクトを設けるようにしている。従って、本発明により、1つの制御オブジェクトまたはそのインスタンスと他の制御オブジェクトまたはそのインスタンスとの間でプロパティ、メソッドおよびイベントの全てを含んだ双方向の通信が可能となる。このため、制御オブジェクトとして提供された機能を十分に活用した制御システムを簡単に構築できる。

50 【0008】さらに、複数のイベントが用意されている

場合に、第1の制御システムのイベントの第1の配列と、第2の制御システムの第2の配列とが一致しなくてもイベントの受渡しが可能のように、これらの配列を変換する手段を設けておくことが望ましい。

【0009】本発明に係る複数の制御オブジェクトおよびそれらのインスタンスを備えた制御システムにより、コンピュータの機種やそれに接続される周辺装置に対しオープンな制御システムを構築できる。例えば、第2の制御オブジェクトを複数の周辺装置の中の第1の周辺装置に対応して用意することによって、第2の制御オブジェクトに第1の周辺装置に固有な仕様に対応した属性値を含んだプロパティと、命令を含んだメソッドと、さらに、第1の周辺装置において非同期に発生するアクションを反映可能なイベントを設定することができる。そして、第2の制御オブジェクトのインスタンスと第1の制御オブジェクトあるいはそのインスタンスをインタフェースオブジェクトあるいはそのインスタンスを介して接続できる。従って、第1の制御オブジェクトあるいはそのインスタンスにおいて第2の制御オブジェクトのインスタンスのプロパティ、メソッドおよびイベントの少なくともいずれかを複数の機種、メーカーあるいは複数の種類の周辺装置に対して普遍的に規定された仕様に変換できる。このため、周辺装置に依存しない共通のアプリケーションプログラムインタフェース（API）をアプリケーションプログラムや上位の制御オブジェクトおよびそのインスタンスに対し提供することができる。さらに、第1の制御オブジェクトのAPIをコンピュータの機種に依存しない普遍的なものにすることも可能であり、機種依存性がなく、汎用性の高いアプリケーションプログラムとすることができる。従って、本発明の制御システムにより、ユーザーは自己の環境に合わせたアプリケーションおよびハードウェアによってPOSシステムなどの周辺装置を多く使用するシステムを簡単に、そしてフレキシブルに構築できる。

#### 【0010】

##### 【発明の実施の形態】

【OLEオートメーションをベースにした例】以下において、マイクロソフト社の提供するプログラム開発環境であるMicrosoft Foundation Class (MFC) 環境下において本発明を実施した例に基づきさらに詳しく説明する。

【0011】MFCはOLEのプログラミングを容易に行えるようにさまざまなライブラリーを提供しており、オブジェクト指向のプログラミング言語であるMicrosoft社が提供するVisual C++（以下VC++）などによってOLEオートメーション機能を用いたシステムの開発を行うことができる。本例では、図1に示すように、アプリケーション（コンテナアプリ）20と、システム内に共通に活用できるように用意されたコモンオブジェクトあるいはそのインスタンス（OCX）10および11とを

階層的に結合して構築された制御システムについて先ず説明する。

【0012】コモンオブジェクトは、他の実行型のソフトウェアを構成するオブジェクトとして提供されても良く、あるいはダイナミックリンクライブラリ（DLL）として提供されるオブジェクトであっても良い。コモンオブジェクトを提供するこれらの実行型のソフトウェアやDLLはEXEサーバーあるいはDLLサーバーと呼ぶことができ、これらに対し、コンテナアプリ20をクライアントあるいはコントロールと呼ぶことができる。OLEに対応したこれらのコモンオブジェクトは、例えば、MFCが提供しているCCmdTargetクラスから派生させることができる。CCmdTargetクラスは、コモンオブジェクトを管理するためのIUnknownインタフェースや、その他のコモンオブジェクトとして必要なインタフェース等をサポートするクラスである。CCmdTargetクラスはコントローラ側からデータを保持した構造体などを受け渡すためのIDispatchインタフェースもサポートしており、このメンバ関数Invokeをコールすることによってコモンオブジェクト側のプロパティあるいはメソッドを用いることができる。

【0013】プロパティは、コモンオブジェクトの属性であり、カラー、テキスト、番号、フォントあるいはプッシュボタンがプッシュされた時の動作などが含まれる。メソッドは、コモンオブジェクトにインプリメントされた、例えば編集機能などのファンクションであり、メソッドをコールすることによってコモンオブジェクトの機能を操作することができる。コモンオブジェクトは、プロパティやメソッドを提供する機能に加え、コモンオブジェクトに対する外部からのアクション、例えばマウスボタンのクリックやキー入力、その他の非同期に発生するアクションをイベントとして発信する機能を備えることができる。これは通常、コントロール側がプロパティ等を用いるのと同様に、コモンオブジェクトにおいてInvoke関数をコールすることによって行われる。このとき、イベント用の結合点が有効なインタフェースを保持していないとイベントの発生は無視されてしまう。イベントの結合点を実現するインタフェースはIConnectionPoint12である。

【0014】さらに、本発明においては、コモンオブジェクトにイベントを受けるためのインタフェースとしてイベント用のIDispatch13を設けてある。従って、OCXの間でイベントの授受を行う場合は、イベントを発信するOCXのインタフェースはIConnectionPoint12に対し、イベントを処理するOCXのイベント用のIDispatch13をアドバースする、すなわちアドレス等の識別子を渡し、コネクションを張れば良い。もちろん、MFCの環境下以外においては、その他の識別子であっても良く、また、インタフェース12または13のいずれ

に側に対して他方の識別子を渡しても良い。

【0015】図1には、コンテナアプリ20があるコン  
モンオブジェクトあるいはそのインスタンスを用いている  
制御システムを示してある。本図に示したシステムで  
は、コンテナアプリ20が第1のオブジェクトのインス  
タンス(OCX)10を作成し、さらに、その第1のOC  
X10が別の第2のオブジェクトのインスタンス(OC  
X)11を作成して制御する制御システムを示してある。  
MFCライブラリーには、コンテナアプリやOCX  
等からInvoke関数をコールする複雑さをカバーするため  
にCOleDispatchDriverクラスが用意されており、OCX  
10はこのクラスから派生させたオブジェクトのメンバ  
ー関数をコールすることにより、他の制御オブジェクト  
のインスタンス、例えば、OCX11を作成しOCX1  
1のメソッド等を活用することができる。

【0016】サーバー内のコンモンオブジェクトを用いる  
場合、例えば、コンテナアプリの動作しているプロセッ  
サーと異なるプロセッサでそのコンモンオブジェクトが  
動作しているケースも考えられる。このようなケースで  
は、本例のような制御システムを、コントロール側とサ  
ーバー側で通信処理を行うことにより構築することがで  
きる。また、1つのプロセッサに活用するコンモンオブ  
ジェクトをコピーして制御システムを構築することも可  
能である。いずれの場合も、用いられるコンモンオブジ  
ェクトのコード自体には変更は全く加えられず、そのコ  
モンオブジェクトを活性化する毎にデータやスタックなど  
の領域が用意され、それを用いてコンモンオブジェクト  
(サーバー側)とコントロール側が通信しながら制御シ  
ステムを構築する。従って、同一のコンモンオブジェクト  
をコントロール側で複数用いることも可能であり、この  
場合、コンモンオブジェクトが複数コピーされるのではな  
く、データやスタックなどの領域がそれぞれに用意され  
る。このようにコンモンオブジェクトが他のオブジェクト  
等によって活用可能な状態になることを本明細書ではイ  
ンスタンスと称する。

【0017】さて、第1のOCX10がコントロール側  
となって第2のOCX11を呼び出して使用したい場  
合、VC++のクラスウィザードを用いてCOleDispatchDriv  
erクラスから派生されたインタフェースオブジェクトを  
用いることが可能であることは上述した通りである。な  
お、本明細書においては、インタフェースオブジェクト  
とは、インタフェースオブジェクトおよびそのインスタ  
ンスを含む概念で用いている。VC++2.0のCOleDispatch  
Driverクラスは、メソッドやプロパティをIDispatchイ\*

```
class COcxDispatchDriver:public CCmdTarget,public COleDispatchDriver
{
    ....
};
```

CCmdTargetクラスは上述したように、サーバーとしての  
機能を備えたオブジェクトを派生するクラスであり、ま  
た、COleDispatchDriverクラスは、第2のOCXに対  
してクライアントあるいはコントロール側として動作し、

\*インタフェースを介して提供する機能はサポートしてい  
るが、第2のOCXが発信するイベントをとらえる機能を  
サポートしていない。従って、コンテナアプリ20の側  
が第1のOCX10の用いる第2のOCX11を識別  
し、第2のOCX11のイベントを捕らえる機能をサポ  
ートしている場合などの特殊なケースを除いて第2のOC  
X11で発生したイベントをとらえることができない。

【0018】本例においては、OCX間の双方向通信を  
可能とするためにOCXにイベント用のインタフェース  
13を設けると共に、COleDispatchDriverクラスからイ  
ベント用のインタフェースを結合できるインタフェース  
オブジェクトを派生させている。すなわち、本例の制御  
システムは、上述したイベントを発信するインタフェー  
スであるIConnectionPoint12とイベントを受信するイ  
ンタフェースであるイベント用のIDispatch13を備え  
たOCX10と、その下流に位置するイベントを発信す  
るインタフェースであるIConnectionPoint12を備えた  
OCX11を有しており、これらをインタフェースオブ  
ジェクト15により接続している。本例のインタフェー  
スオブジェクト15は、OCX11のプロパティおよび  
イベントをOCX10に提供する機能に加え、OCX1  
1のIConnectionPoint12にOCX10のイベント用の  
IDispatch13のアドレスを渡す機能を備えるようにし  
ている。このため、OCX10が本例のインタフェース  
オブジェクト15をインスタンスとして作成し、さら  
に、インタフェースオブジェクト15がOCX10に基  
づきOCX11を作成すると、OCX11のIConnectio  
nPoint12とOCX10のイベント用のIDispatch13  
の間にコネクションが張られ、OCX11の発信したイ  
ベントがOCX10によって受けられるようになる。従っ  
て、OCX11とOCX10との間で双方向の通信が可  
能となり、コンテナアプリ20の側は第2のOCX11  
を認識する必要は全くなくなる。また、第2のOCX1  
1もコンテナアプリ20に左右されないオブジェクトと  
して提供することができる。

【0019】本例においては、インタフェースオブジェ  
クト15は、COcxDispatchDriverクラスから派生されて  
いる。このCOcxDispatchDriverは、CCmdTargetクラス  
と、COleDispatchDriverクラスとを多重継承(Multiple  
Inheritance)して派生したオブジェクトクラスであり、  
次のようになる。

【0020】

プロパティ/メソッドに対するアクセスを簡単に行なえ  
る機能を備えたオブジェクトクラスである。従って、ク  
ラスウィザードにより上記の2つのクラスから派生させ  
たオブジェクトクラスであるCOcxDispatchDriverクラス

は、上記の2つの主な機能を備え、これらに加えそれぞれのクラスの機能を全て継承し、サポートされたオブジェクトクラスなので、本発明に係るインタフェースオブジェクト用のオブジェクトクラスとして適している。CCmdTargetクラスとCOleDispatchDriverクラスには重複す\*

\*る部分がないため、全く問題なく多重継承が可能である。

【0021】このように派生されたCOcxDispatchDriverクラスの仕様は以下の通りである。

【0022】

```
Class COcxDispatchDriver : public CCmdTarget,
                           public COleDispatchDriver
COcxDisptchDriver::COcxDispatchDriver() // 生成
public: // メンバ変数
IID m__IIDEvents; // イベント インタフェース ID
UNIT m__nEvents; // イベント数
CDWordArray m__dispID // ユーザが定義したディスパッチマップに対応して
                        ディスパッチIDを変換するための配列
public: // メンバ関数
Bool PrecreateDispatch(
    REFCLSID clsid,
    CStringArray& EntryNames,
    CDWordArray& DispIDs,
    CStringArray& ParamInfo,
    COleException* pError=NULL); // オブジェクトインスタンス
                                を作成し、インタフェースの情報を格納する。
```

【0023】

```
Bool EstablishConnection(
    LPUNKNOWN pUnkSink,
    LPCDMENTRY pDispMap,
    CStringArray& EntryNames,
    CDWordArray& DispIDs,
    CStringArray& ParamInfo,
    COleException* pError=NULL); // イベントのインタフェース
                                が受側と発行する側とで一致しているかを確認し
                                双方を結合する。
```

【0024】

```
void DestroyConnection(); // コネクションを開放し、オブジェクト
                            インスタンスを削除する
```

本例のCOcxDispatchDriverクラスから派生されるインタフェースオブジェクトは、OCX双方のイベントのインタフェースのコネクションを張る機能を備えているだけなのでイベントの処理関数(イベントハンドラ)は含んでいない。イベントハンドラは、イベントを受け入れるOCXの側に用意される。まず、イベントを発行する側のOCXが返すイベントの外部名に合わせてディスパッチマップを記述する。ディスパッチマップの各イベントの順番は任意で良く、後述するようにイベントとのコネ※

※クションをセットするときに実際に受信するイベントの順番との対応付けがなされる。ただし、OCXの全部のイベントを完全に記述しておく必要がある。

【0025】ディスパッチマップのエントリは例えば以下のようになり、マクロのパラメータは、順に、ディスパッチドライバクラス名、外部イベント名、イベントハンドラ名、戻り値、パラメータ情報となる。

【0026】

```
DISP __FUNCTION( __DSoprn, "ControlCompleteEvent", ControlCompleteEvent
```

```
VT __EMPTY, VTS_I4 VTS__SCODE VTS __PBSTR VTS __I4)
```

このような外部イベント名に合わせたディスパッチマップを記述することにより、OCXで発生したイベントを直観的に判りやすい形式で受け取ることができる。さらに、イベントハンドラのプロトタイプ宣言は下記のように

になるので、通常の間数の形式で記述でき、プログラム開発が極めて容易となる。

【0027】



```
void __DSoprn::ControlCompleteEvent(long ControlID, SCODE Result,
    BSTR FAR*, pString, long data);
```

次に、C0cxDispatchDriverクラスから派生されたインタフェースオブジェクト15によって第1のOCX10と第2のOCX11とのコネクションを設定する処理を図2および図3に示したフローチャートに基づき説明する。

【0028】図2に示すように、インタフェースオブジェクトのインスタンス15が生成され、PrecreateDispatch がコールされると、ステップ31において、clsid によって指定された第2のOCXであるオブジェクトインスタンス11が生成され、動作を開始する。変数clsid は作成するOCXを識別するための128ビットの値が格納されている。ステップ32において第2のOCX11が動作を開始すると、ステップ33においてこのOCX11のIDispatch インタフェースを取得し保存する。インタフェースオブジェクト15は、このIDispatch を通じてInvoke関数をコールし、インスタンス11にプロパティまたはメソッドを渡すことができる。ステップ34において、第2のOCX11にプロパティあるいはイベントを渡すインタフェースが確認されると、ステップ35において、第2のOCX11のイベントのインタフェースがチェックされる。

【0029】ステップ36において、イベントに関するインタフェースがあるとステップ37において、イベントに関する情報を格納する。第2のOCX11において用意されているイベント名リストはEntryNamesに、ディスパッチIDリストはDispIDs に、また、パラメータ情報リストはParamInfo に格納される。ステップ38においてこれらの処理が通常に終了すると次に図3に示したイベントの対応付けを行う工程に移行する。一方、コネクションがセットできない場合はステップ39においてエラー処理を行う。

【0030】図3にイベントの対応付けを行う工程を示してある。pUnkSinkにイベントの受入れ側となるOCX10のインタフェースの識別子（本例ではOCX10のイベント用IDispatch のアドレス）が設定され、EstablishConnection がコールされる。まず、ステップ41においてカウンタfcntをゼロクリアする。次にステップ42においてカウンターfcntを確認し、ステップ43において第2のOCX11から得られたイベント名EntryNames (fcnt)がOCX10のイベントハンドラに予め用意したマップpDispMapにあるか否かをチェックする。マップpDispMapにある場合は、さらに、ステップ44において、パラメータ情報ParamInfo (fcnt)がイベントハンドラに予め用意したマップpDispMapと一致するかを確認する。確認できた場合は、ステップ45において、第2のOCX11から得られたイベントのディスパッチIDであるDispIDs (fcnt)をイベントハンドラに予め用意したマップpDispMapでのインデックスに変換するための配列

m \_\_dispIDを設定する。ステップ46において、カウンタfcntをインクリメントし、すべてのイベントに対し、これらのステップを繰り返す。予め用意したマップpDispMapとの対応が付くと、ステップ47において、pUnkSinkにある第1のOCX10のインタフェースの識別子を第2のOCX11のイベントに関するインタフェースに渡しコネクションを張る。本例では、MFC環境下にあるので、IConnectionPointにアドレスを渡し、イベントに関するコネクションをセットする。これによって、第2のOCX11においてInvoke関数をコールすると、イベントがOCX10に渡される。すなわち、第1のOCX10が第2のOCX11からイベントを受け取れるようになる。

【0031】ステップ48においてコネクションが張られると、ステップ49においてイベントのディスパッチIDを変換するための配列m \_\_dispIDを保存する。ステップ50において、上記のステップが全て正常に終了するとインタフェースオブジェクト15と第2のOCX11とのコネクションのセットが終了し、第1のOCX10に対し第2のOCX11のプロパティあるいはメソッドを提供できるようになり、さらに、第2のOCX11から第1のOCX10にイベントが伝えられるようになる。

【0032】このように、コモンオブジェクトにイベントを受け取るためのインタフェースを用意し、さらに、インタフェースオブジェクト15によってインタフェース間のコネクションを張れば、コモンオブジェクトあるいはそのインスタンス同士の間で双方向の通信が可能となる。従って、コモンオブジェクトあるいはそのインスタンスを用いて処理の速いフレキシブルな制御システムを構築することができる。さらに、本例のインタフェースオブジェクトにおいては、イベントに関するコネクションをセットするときに、イベントを発信する側のイベントの配列と、イベントを受け入れる側のイベントの配列を確認し、対応付けるようにしている。従って、第1のOCX10に対し、第2のOCX11のインタフェースが完全に既知となっていなくとも、イベントの種類と数が合致していればコネクションをセットすることができる。すなわち、プロパティ、メソッドおよびイベントの名称が合致していれば複数のコモンオブジェクト間に双方向のコネクションをセットすることができる。このため、コモンオブジェクトの開発が容易となり、バージョンアップが行われたり、異なったオブジェクトが採用された場合であっても、制御システムをフレキシブルに、そして、安全、確実に構築することができる。

【0033】さらに、イベントを受け取ったときに起動されるイベントハンドラも通常の関数の形式で記述することができるので、オブジェクトの開発が極めて容易と

なり短時間でできるようにする。

【0034】図4に、第2のOCX11から、インタフェースオブジェクト15に用意されたIDispatch インタフェースのイベントを通知する呼び出し Invoke(EventID,...) がコールされた場合の処理を示してある。IDispatch インタフェースは Invoke() および、それをサポートする基本的な関数 (AddRef(), Release(), QueryInterface(IID&, LPUNKNOWN\*) 等) を提供しており、IDispatch インタフェースが、相手のOCXが発生するイベントの IID (Interface ID) にQueryInterface(...) を介して応答する。Invoke() がコールされると、この時点でパラメータとしてイベントの種類を規定するdispatch ID やイベントのパラメータを保持する構造体などがOCX10に渡される。イベントを受け入れる側のOCX10では、ステップ60においてパラメータが正当か否かをチェックする。正当な場合はステップ61において、第2のOCX11から得られたイベントの配列dispIDMemberを用意されたイベントハンドラのマップpDispMapに合わせて、変換用の配列m\_\_dispIDに基づき変換する。ステップ62において、dispIDMemberに対応するイベントハンドラをpDispMapから取り出す。そして、ステップ63において、イベントハンドラが有効か否かを確認し、有効な場合はステップ64においてイベントに対応するイベントハンドラを呼び出し、イベントハンドラに定義された処理、例えば、アプリケーション側にイベントを発信するなどの処理を行う。

【0035】なお、上記では、OCX10に1つのOCX11が繋がる場合を例に説明しているが、複数のOCX11と繋げることももちろん可能である。イベントを受け入れるインタフェースが単独の場合は、同一のイベントの配列である必要があるため、OCX11のタイプは同一であることが望ましい。

【0036】このように、本発明のシステムにおいては、複数のコモンオブジェクトおよびそれらのインスタンス、そしてインタフェースオブジェクトを用いることにより、フレキシブルな制御システムを簡単に構築することができる。それぞれのコモンオブジェクトには、活用する相手のコモンオブジェクトに関し最小限の情報を用意しておけば良く、また、活用される相手のコモンオブジェクトに関する情報も最小限で済む。従って、個々のコモンオブジェクトのカプセル化をいっそう進めることができる。その一方で、本発明のシステムでは、OCX同士の間でイベントも含めた双方向の通信が可能であり、活用する相手のOCXを完全に動作させることができ、それらの備えた機能を最大限に発揮させることができる。

【0037】なお、本例においては、第1および第2のOCXからなる2層の制御システムを用いて説明しているが、3層あるいはそれ以上の階層構造をもった制御システムであっても同様に構築することができる。また、

上記においては、マイクロソフト社の提供するMFC環境下において本発明のオブジェクトインタフェースを実現した例を用いて説明しているが、MFCライブラリを用いなくとも、同等の機能を備えたオブジェクトインタフェースを作成することは可能である。さらに、共通して活用できるオブジェクトが用意されたシステム環境であれば、マイクロソフト社のOLEオートメーションでなくとも上記と同様のコモンオブジェクトを用いたシステムを構築できる。このようなコモンオブジェクトを用いたシステムは、1つのプロセッサを備えたコンピュータ単体で実現することもでき、また、ネットワーク環境で接続された複数のプロセッサを有するシステムにおいても構築できることは上述した通りである。

【0038】また、本例では、インタフェースオブジェクトを採用することによって、より拡張性が高く、また、プログラム開発の容易な制御システムをしているが、インタフェースオブジェクトの機能と第1のOCXの機能を備えたコモンオブジェクトを提供することももちろん可能であり、そのようなOCXを用いたシステムによってもイベントを含めた双方向通信の可能な制御システムを構築できることはもちろんである。

【0039】〔周辺装置の制御システム〕図5に、本発明に係るインタフェースオブジェクトと、複数のコモンオブジェクトおよびそのインスタンスによって周辺装置の制御システムを構築した例を示してある。図5にはパーソナルコンピュータ (パソコン) 70を中心に構成されたPOSシステムを示してある。パソコン70にはPOSアプリケーションプログラム71がインストールされており、パソコンのオペレーティングシステム (OS) 105の上で動作する。OS105は、キーボードドライバ106やモニタディスプレイドライバ107等を介してパソコンとして通常必要な周辺装置を制御する機能を備えており、アプリケーションプログラム71とキーボードあるいはモニタディスプレイ (不図示) との間のデータ転送はOS105を介して行われる。

【0040】POSシステムには、パソコンに通常用いられるこれらの周辺機器に加えて、客先に金額等を表示するカスタムディスプレイ110、レシート等の印刷を行うレシートプリンタ112、チェックなどの印刷を行うスリッププリンタ113、さらに、金銭を保管するキャッシュドロワ115が必要となり、これらの周辺装置はRS-232Cポート等の拡張用のポートに接続される。例えば、カスタムディスプレイ110がRS-232Cポートに接続され、カスタムディスプレイ110をパススルーしてレシートプリンタ112およびスリッププリンタ113を備えたプリンタ111が接続される。キャッシュドロワ115は、プリンタ111の下部に設置され、プリンタ111の制御機構を介して操作される。これらの周辺装置は多くのメーカーから様々な機種が市販されており、ユーザーは自己の環境に適したもの

を選択してPOSシステムを構築可能である。しかしながら、メーカーや機種異なる周辺装置は個々に仕様異なるので、市販されている全ての周辺装置に適合したアプリケーションプログラムの作成は不可能である。さらに、周辺装置がバージョンアップされると、これに伴い仕様も変更になる。従って、従来は、ユーザーは自己に都合の良い周辺機器によってPOSシステムを構築することが困難なこともあり、さらに、周辺機器がバージョンアップされても新しい機種を構築済のPOSシステムにすぐに適用できるとは限らなかった。

【0041】これに対し、上述したOCXによって制御システムを構築すると、非常にオープンなシステムを形成できる。従って、どのような機種の周辺機器を用いたPOSシステムであっても簡単に構築できる。また、周辺機器のバージョンアップに対しても簡単に対処できる。例えば、図5に示す本例の周辺機器の制御システム72は、3つのレベルのOCXを備えている。第1のレベルのOCXとして、レシートプリンタフォーマット変換用OCX73と、スリッププリンタフォーマット変換用OCX74が用意されている。これらのOCX73および74では、例えば、アプリケーションプログラム71から送られた売上品リストや合計金額等のデータを所定のフォーマットに配置する処理が行われる。所定のフォーマットは、OCX73あるいは74の内部に設定されていても良いし、下位レベルの制御用OCXであるレシートプリンタ制御用OCX75あるいはスリッププリンタ制御用OCX76がプロパティとして有しており、これらの制御用OCXのプロパティを得て変換用OCX73あるいは74がフォーマットを設定しても良い。いずれにしても、アプリケーションプログラム71は、出力表示されるフォーマットに関係なく、出力用のデータを変換用OCX73あるいは74に受け渡せば良い。従って、インタフェースの形式を限定でき、汎用性の高いアプリケーションプログラムとして提供することができる。また、アプリケーションプログラムから固有のフォーマットで出力されたデータを、第1レベルのOCXによって下位のOCXに共通するフォーマットに変換することも可能である。このように、OCXを用いて個別に開発されたアプリケーションプログラムの汎用性を高めることも可能である。

【0042】第2のレベルのOCXとしてレシートプリンタ制御用OCX75、スリッププリンタ制御用OCX76、キャッシュドロワ制御用OCX77およびカスタムディスプレイ制御用OCX78が用いられている。これらのOCX75～78は、アプリケーションプログラムあるいは上位のOCXに対し、所定の仕様のインタフェース(API)を提供するOCXである。従って、POSシステムを構成するプリンタ等の周辺装置のメーカーや機種に係わりなく、アプリケーションプログラムや上位のOCXは所定の仕様でデータを提供すれば良い。

このレベルのOCX75～78は、周辺装置固有の仕様が反映された下位のドライバレベルのOCXのプロパティを得て、共通の仕様で入力されたデータを下位のOCX、すなわち、実際にシステムを構築している周辺装置の仕様に合わせたデータに変換する。

【0043】第3のレベルのOCXは、プリンタドライバOCX91および92と、キャッシュドロワドライバOCX93、およびディスプレイドライバOCX94である。これらのOCX91～94は、個々の周辺装置に対応したコモンオブジェクトのインスタンスであり、通常はメーカー毎あるいは機種毎に異なり周辺装置と共に提供される。これらのドライバレベルのOCX91～94は、例えば、最大印字行数や印字行ピッチと言ったプリンタ固有の仕様や設定状態(プリンタステータスと称する)をプロパティとして備えており、これらのプロパティは上位のOCXやアプリケーションプログラムで参照できるようになっている。また、指定された位置に文字列を印字する、すなわち、印字命令を出力するというメソッドや、プリンタステータス等のプロパティなどを備えている。従って、ドライバーレベルである第3レベルのOCXからは、例えば、プリンタ用のOCXであれば、送られた印字位置および印字文字列のデータに基づき、改行量(すなわち行ピッチ)、改行コマンド、印字データおよび印字コマンド、オートカットコマンドが所定の順序でポートドライバ100を介してプリンタに送信される。

【0044】第3レベルのOCXは、プリンタから送信される処理結果およびエラーステータス等の非同期に発生するアクションも受け取る。そして、これらのアクションの内容を対応する周辺装置の仕様に基づき解釈し、普遍的な形式に変換してイベントとして上位のOCXに返す。

【0045】本例の制御システムにおいては、少なくとも第1および第2のレベルのOCXは上述したイベント用をIDispatch インタフェースを備えたOCXである。従って、本発明に係るインタフェースオブジェクト81a、81b、82a～82dを用いて上位のOCXと下位のOCXを接続することにより、プロパティやメソッドに加え、イベントを伝達するインタフェースを上位および下位のOCXの結ぶことができる。従って、第2レベルの制御用OCX75～78は第3レベルのドライバクラスのOCX91～94の発行するイベントを受けて、そのまま、あるいはさらに普遍的な形式に変換し上位のOCXやアプリケーションプログラムに伝達する。これに基づき、アプリケーションシステムや上位のOCXは、ユーザーに対してメッセージを出力したりエラー処理ルーチンを起動するなどの処理を行う。従って、本例の制御システムにおいては、非同期に発生するアクションに対して迅速にそしめる的確な処理を行うことができる。

【0046】このように、本例の制御システムは、OCXを用いているのでユーザーの採用する周辺装置に適合したカスタマイズが容易に行えるシステムであり、また、OCX間でプロパティ、メソッドおよびイベントの双方向の通信が確保されているので、処理速度が早くて正確に処理が行える制御システムである。

【0047】レシート印字機能を例にとりてさらに詳しく説明する。第3レベルのOCXであるプリンタドライバOCX91は、レシートプリンタ112のプリンタステータスをプロパティとして備えている。また、プリンタ112に印字命令を出力するメソッドを備えている。さらに、プリンタ112からリアルタイムで送られてくる用紙なし（ランアウト）やカバーオープンといったエラーステータスに対応して非同期にイベントを発生する。

【0048】アプリケーションプログラム71からデータが第1レベルのレシートプリンタフォーマット変換用OCX73に渡され、所定のフォーマットに変換されたのち、第2レベルの制御用OCX74に渡される。そして、制御用OCX74がドライバOCX91の印刷を実行するメソッドをコールすると、ドライバOCX91がレシートプリンタ112のプリンタステータスに合致した適当なコマンドおよびデータをレシートプリンタ112に対し送信し、レシート印字を実行する。レシートプリンタ112は、送信されたコマンドを実行し、所定のステータスをドライバOCX91に返す。ドライバOCX91は、返されたステータスを解釈し、エラーステータスがアクティブでなければイベントを発生せず、印刷処理は終了する。また、上位のOCXであるレシートプリンタ制御用OCX75は、プリンタステータスのプロパティを参照することによってプリンタ処理の結果を知ることもちろん可能である。

【0049】一方、印刷実行のコマンドを実行した結果、あるいは待機中にレシートプリンタ112にエラーが発生してエラーステータスがアクティブになると、プリンタ112はエラーステータスをドライバOCX91に送信する。この場合、ドライバOCX91はイベントを発生し、エラーの発生を上位の制御用OCX74に伝達する。制御用OCX74は、イベントに対応して特定の処理を行うことも可能であるし、さらにフォーマット変換用OCX74およびアプリケーションプログラム71にイベントを通知し、注意を喚起することによって所定の処理を行わせることも可能である。

【0050】これらのプロパティ、メソッドおよびイベントに加え、スリップリンタ113に対応したドライバOCX92としては、スリッ印字に特有のプロパティおよびイベントを設けておくことが望ましい。例えば、プロパティにスリッ用紙の有無を加えることによって、上位のOCXやアプリケーションが印刷実行を指示するタイミングの判断が可能となる。また、イベント

に用紙挿入検出、用紙終端検出を加えることにより、アプリケーションプログラムなどにスリッ印刷後の処理やエラー処理をスムーズに行わせることができる。

【0051】キャッシュドロー制御用OCX77はキャッシュドロードライバOCX93と通信する。キャッシュドロードライバOCX93はキャッシュドローオープン・クローズを含めたキャッシュドローステータスをプロパティとして備えている。また、イベントとしてキャッシュドローエラーやキャッシュドローオープン検出を発生することができる。

【0052】カスタマディスプレイ制御用OCX78は、ディスプレイドライバOCX94と通信し、このドライバOCX94は、表示位置や表示文字列を指定した表示命令をメソッドとして備えている。また、プロパティとしては表示桁数や表示色などのディスプレイ仕様を備えており、ディスプレイ制御用OCX78はこのプロパティに沿って表示データを与える。また、カスタマディスプレイはイベントに該当するエラー等の発生は少ないので、イベントを発生する機能を削除することも可能である。

【0053】このように、本発明の制御システムは、複数のOCXを用いて階層的な構造を備えており、エラー等を含めた非同期に発生するアクションに対しても迅速に対処可能な制御システムである。また、本例の制御システムは、パソコンに接続される周辺装置に対しフレキシブルに対応できるオープンな制御システムである。例えば、プリンタ111は、レシート印字機能112と、スリッ印字機能113と、さらにドロー115の制御機能とを備えており、制御システムはそれぞれの機能毎にOCX91～93が用意されている。このプリンタ111に対し、スリッ印字機能の仕様（例えば、最大印字桁数、実行可能なコマンドセット等）の異なるプリンタに置き換える場合は、スリッ印字機能を制御するドライバ用のコモンオブジェクトを新しいプリンタの仕様に対応するものに置き換えれば良い。周辺装置の制御システムにおいては、置き換えられた新しいドライバ用のコモンオブジェクトからOCXを作成し新しいプリンタに適合した制御システムを自動的に形成する。プリンタがバージョンアップされて仕様が変わった場合でも、同様にドライバ用のコモンオブジェクトを置き換えるだけで良く、その他のコモンオブジェクトやアプリケーションプログラムに変更を加える必要はない。

【0054】カスタマディスプレイについても同様であり、仕様変更された場合等はドライバ用のコモンオブジェクトを変更すれば良い。また、バーコードリーダ等の他の周辺装置を加えてPOSシステムが構成される場合は、それに対応した制御用のコモンオブジェクトとドライバ用のコモンオブジェクトがパソコンのシステム内に用意されておれば、アプリケーションあるいは上記の

OCXによって、下位のOCXが作成され、バーコードリーダ等を含めた周辺装置の制御システムが形成される。

【0055】同様に、アプリケーションプログラムとのインタフェースを担当する上位のOCXがパソコンのハード等に起因する仕様の違いを吸収し、アプリケーションに対し共通のインタフェースを与えるものであれば、パソコンのハード等の相違とは無関係に共通のアプリケーションプログラムを用いてシステムを構築することも可能となる。パソコン内に用意されたコンモノブジェクトから制御システムがアプリケーションプログラムによって作成され、制御システムを構成するOCXによってパソコンおよびそれに接続された周辺装置に適合したシステムが自動的に形成される。このように、本発明のOCXを用いた制御システムはオープンなシステムであり、インタフェースの仕様を予め合意しておけば、アプリケーションを供給する側は、パソコンやそれに接続される周辺機器の仕様および接続方法等は一切考慮せずにアプリケーションを開発し供給することができる。従って、短期間で開発が可能となり、安価に提供できる。また、ユーザーもパソコンや周辺装置に係わりなく、自己の目的や環境に適したアプリケーションを自由に選択することができる。

【0056】一方、周辺装置を提供する側も、供給する周辺装置の仕様に対応したコンモノブジェクトを用意し、例えば、周辺装置と共に供給することによって、パソコンやアプリケーションに限定されない汎用的な周辺装置を提供することができる。従って、ユーザーも自己の目的や環境に適した周辺装置を自由に購入し、システムを構築することができる。

【0057】さらに、本発明に係る制御システムでは、インタフェースオブジェクトによってコンモノブジェクトおよびそのインスタンス同士の間で双方向にプロパティ、メソッドおよびイベントの伝達が可能となっている。従って、本発明の基づく制御システムでは、システムの応答が遅れたり、システムの機能が限定される等のシステムをオープン化する上での弊害は全くない。また、制御システムの機能アップやバージョンアップも個々の制御オブジェクト毎に行うことができるので、システム開発を効率良く、短期間に行える。また、ユーザーもシステムのグレードアップや変更等を簡単に行えるようになる。

【0058】なお、上記では、用いられる周辺装置の種類や数が多く、また、ユーザーの環境に合わせて様々な機種種の周辺装置が用いられる例としてPOSシステムを説明しているが、POSシステムに限定されないことはもちろんである。近年のパソコンを中心としたシステムは、ユーザーの目的や能力に合わせて様々な仕様の周辺装置が組み合わされるようになりつつある。本発明に係る共通する制御オブジェクトを用いた制御システムは、

様々なユーザーの要望に対してフレキシブルに対応可能なシステムであり、POSシステムに限らず、今後様々なシステムに対して適用可能である。

【0059】

【発明の効果】以上に説明したように、本発明においては、アプリケーションプログラム間で共通に活用できる制御オブジェクト、すなわち、コンモノブジェクトおよびそのインスタンスを用いて制御システムを構築しており、さらに、これらコンモノブジェクトおよびそのインスタンスの間で双方向の通信を可能としている。従って、コンモノブジェクトを活用して多種多様なオペレーティングシステムあるいはアプリケーションプログラムを簡単に構築することができる。このように、本発明は、入出力端末や、処理する対象などにより多種多様に変化するコンピュータを中心とした近年のシステムにおいて、いっそうフレキシブルに、そしてカスタマイズ可能なシステムを実現するのに好適な制御システムおよびその構築方法を提供するものである。

【0060】特に、上述したようなコンピュータの機種種やそれに接続される周辺装置の種類および数量がユーザーの環境に応じて多種多様に変化する制御システムを構築するのに本発明は好適である。本発明に係る制御システムを用いることにより、アプリケーションプログラムに対し周辺装置に依存しない共通のアプリケーションプログラムインタフェースを提供可能である。さらに、本発明に係る制御システムは、汎用性の非常に高い制御システムでありながらエラー等の非同期に発生するアクションに対しても迅速にアプリケーションを作動させることが可能である。従って、ユーザーに対し自己の目的に対し最適な快適な動作環境を容易に構築できる制御システムを提供できる。

【図面の簡単な説明】

【図1】本発明の実施例の制御システムの概略構成を示すブロック図である。

【図2】図1に示す制御システムにおいて、インタフェースオブジェクトとオブジェクトインスタンスとの間でコネクションを確認する過程を示すフローチャートである。

【図3】図1に示す制御システムにおいて、インタフェースオブジェクトとオブジェクトインスタンスとの間でイベントの対応付け、コネクションを確立する過程を示すフローチャートである。

【図4】図1に示す制御システムにおいて、オブジェクトインスタンス側からイベントをインタフェースオブジェクトに伝える過程を示すフローチャートである。

【図5】本発明の制御システムによって構築されたPOSシステムの周辺装置を制御するシステムを示すブロック図である。

【符号の説明】

10・第1のOCX

21

22

11・・・第2のOCX

15・・・インタフェースオブジェクト

20・・・アプリケーション

70・・・パソコン

71・・・POSアプリケーションプログラム

72・・・制御オブジェクトによって構成された制御システム

100・・・ポートドライバ

110・・・カスタマディスプレイ

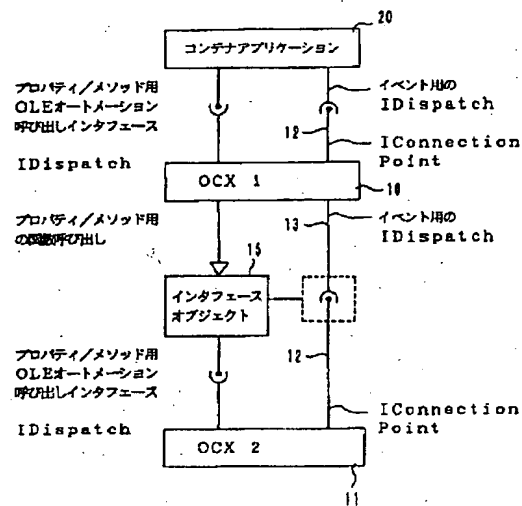
111・・・プリンタ

112・・・レシート印字機構

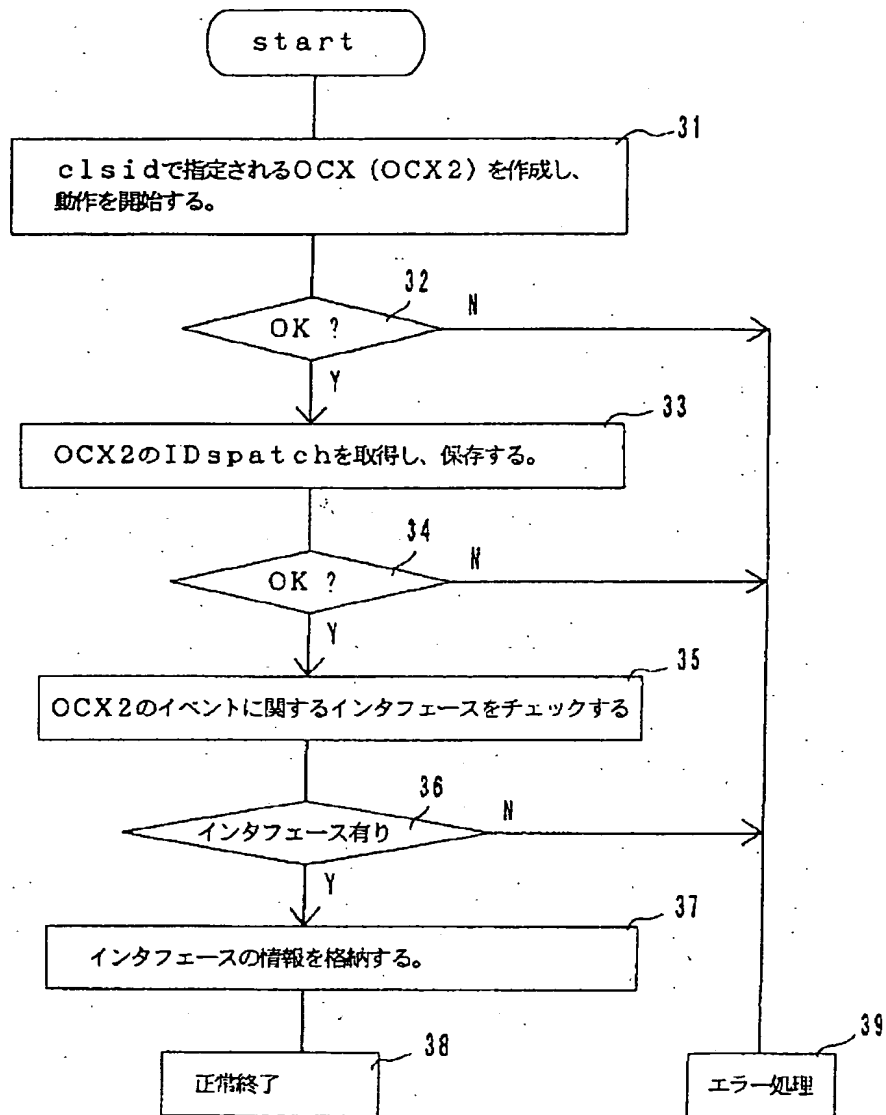
113・・・スリップ印字機構

115・・・キャッシュドロワ

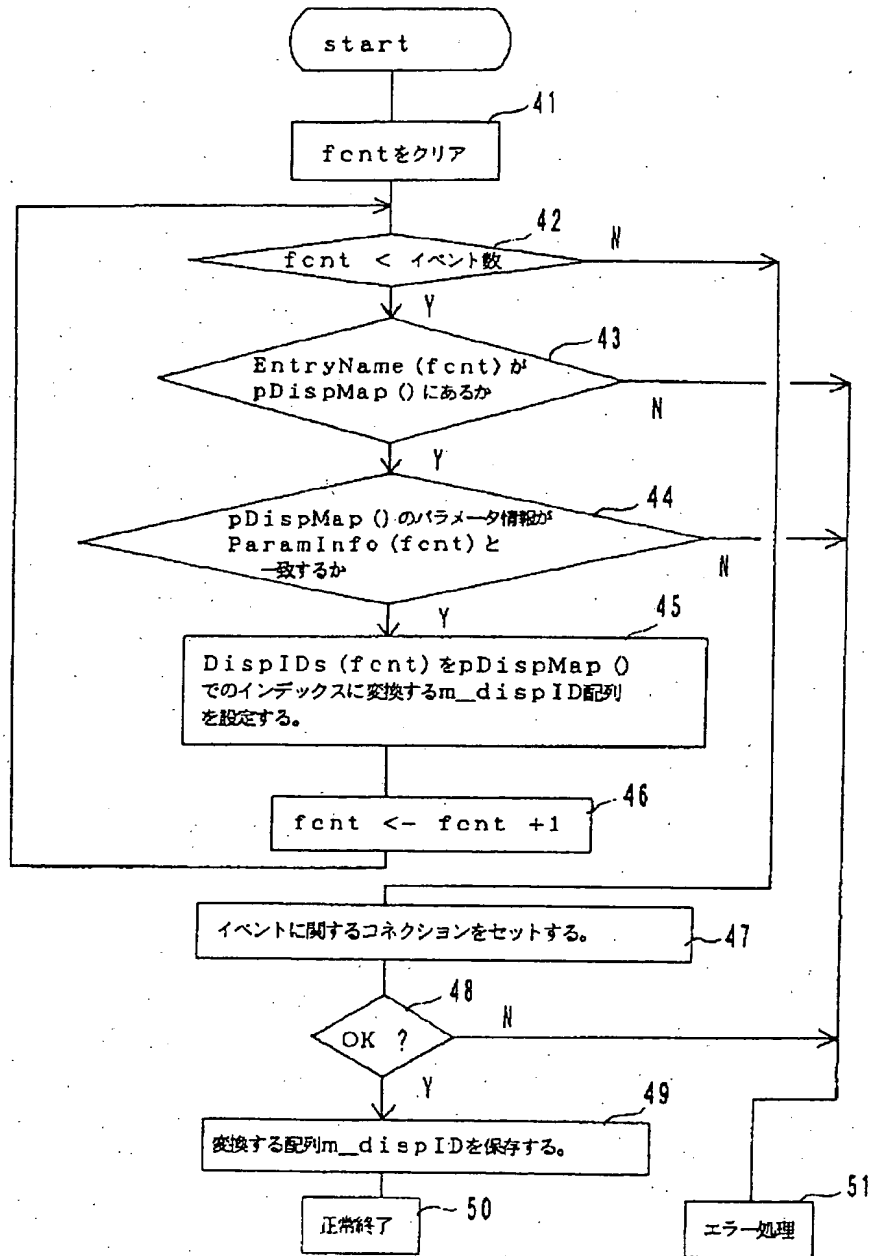
【図1】



【図2】

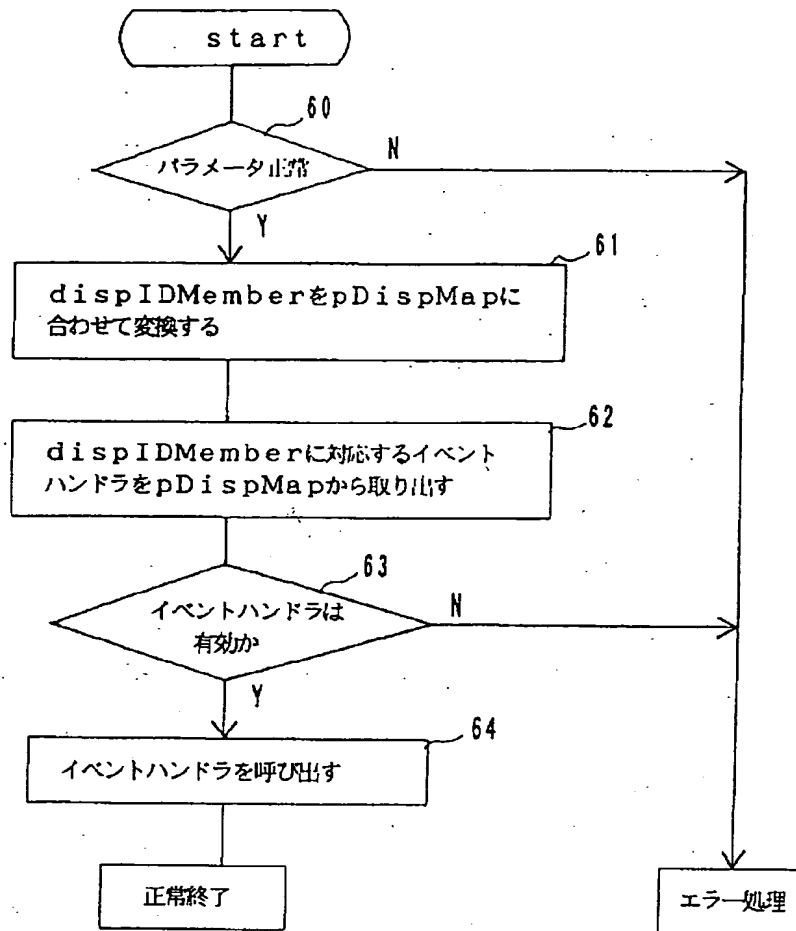


【図3】





【図4】



【図5】

